



US006407748B1

(12) **United States Patent**
Xavier

(10) Patent No.: **US 6,407,748 B1**
(45) Date of Patent: ***Jun. 18, 2002**

(54) **METHOD AND APPARATUS FOR
MODELING INTERACTIONS**

- (75) Inventor: **Patrick G. Xavier, Albuquerque, NM
(US)**
- (73) Assignee: **Sandla Corporation, Albuquerque, NM
(US)**
- (*) Notice: **Subject to any disclaimer, the term of this
patent is extended or adjusted under 35
U.S.C. 154(b) by 0 days.**

**This patent is subject to a terminal dis-
claimer.**

- (21) Appl. No.: **09/062,333**
- (22) Filed: **Apr. 17, 1998**
- (51) Int. Cl.⁷ **G06G 5/00**
- (52) U.S. Cl. **345/672; 345/419; 345/420;
345/427; 345/958; 345/959**
- (58) Field of Search **345/419, 420,
345/424, 427, 438, 958, 959, 672**

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,890,242 A	12/1989	Sinha et al.	364/518
4,944,034 A	7/1990	Ohsawa	364/522
5,014,230 A	5/1991	Sinha et al.	364/578
5,047,916 A	9/1991	Kondo	364/167.01
5,056,031 A	10/1991	Nakano et al.	364/461
5,056,045 A	10/1991	Ohsawa	364/522
5,307,282 A	4/1994	Conradson et al.	364/468
5,347,459 A	9/1994	Greenspan et al.	364/461
5,515,489 A	5/1996	Yaeger	395/152
5,548,694 A *	8/1996	Gibson	345/424
5,572,634 A	11/1996	Dulak, Jr.	395/119
5,613,049 A *	3/1997	Brechner et al.	345/420
5,625,575 A	4/1997	Goyal et al.	374/589
5,675,720 A *	10/1997	Sato et al.	345/419
6,099,573 A *	8/2000	Xavier	345/419

OTHER PUBLICATIONS

Weld et al. "Geometric Representation of Swept Volumes with Application to Polyhedral Objects", The International Journal of Robotics Research, vol. 9, No. 5, Oct. 1990, pp. 105-117.*

Henry Fuchs, University of North Carolina at Chapel Hill; Zvi M. Kedem and Bruce F. Naylor, The University of Texas at Dallas, On Visible Surface Generation by a Priori Tree Structures, 1980 ACM 0-89791-021-4/80/0700-0124.

Chris L. Jackins and Steven L. Tanimoto, Oct-Trees and Their Use in Representing Three-Dimensional Objects Computer Graphics and Image Processing, 14, 249-270 (1980).

Donald Meagher, Geometric Modeling Using Octree Encoding, Computer Graphics and Image Processing, 19, 129-147 (1982).

Bernard Faverjon, Hierarchical Object Models for Efficient Anti-Collision Algorithms, CH2750-8/89/000 1989 IEEE.

Sean Quinlan, Efficient Distance Computation Between Non-Convex Objects, 10-50-4729/94 1994 IEEE.

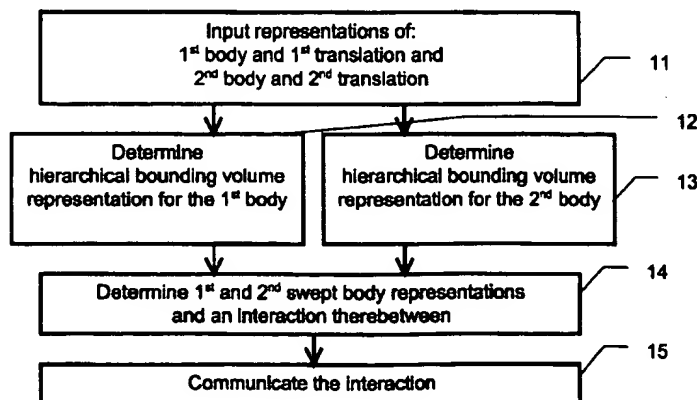
(List continued on next page.)

Primary Examiner—Mark Zimmerman
Assistant Examiner—Kimbinh T. Nguyen
(74) *Attorney, Agent, or Firm*—V. Gerald Grafe

(57) **ABSTRACT**

The present invention provides a method and apparatus for modeling interactions that overcomes drawbacks. The method of the present invention comprises representing two bodies undergoing translations by two swept volume representations. Interactions such as nearest approach and collision can be modeled based on the swept body representations. The present invention is more robust and allows faster modeling than previous methods.

22 Claims, 4 Drawing Sheets



OTHER PUBLICATIONS

Yuichi Sato, Mitsunori Hirata, Tsugito Maruyama, Yuichi Arita, Efficient Collision Detection Using Fast Distance-Calculation Algorithms for Convex and Non-Convex Objects, Proceedings of the 1996 IEEE International conference on Robotics and Automation, Minneapolis, Minnesota—Apr. 1996.

S. Gottschalk, M. C. Lin, D. Manocha, OBBTree: A Hierarchical Structure for Rapid Interference Detection, Computer Graphics Proceedings Annual Conference Series, 1996.

John Canny, Collision Detection for Moving Polyhedra, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-8, No. 2, Mar. 1986.

Stephen Cameron, Collision Detection by Four-Dimensional Intersection Testing, IEEE Transaction on Robotics and Automation, vol. 6, dNo. 3, Jun. 1990.

Stephen Cameron, A Comparison of Two Fast Algorithms for Computing the Distance Between Convex Polyhedra, Submitted to IEEE TR&A as a Communications Paper, Jul. 1996.

Elmer G. Gilbert, Daniel W. Johnson and S. Sathya Keerthi, A Fast Procedure for Computing the Distance Between Complex Objects in Three-Dimensional Space, IEEE Journal of Robotics and Automation, vol. 4, No. 2, Apr. 1988.

Patrick G. Xavier, A Generic Algorithm for Constructing Hierarchical Representations of Geometric Objects, 1996 IEEE International Conference on Robotics and Automation.

* cited by examiner

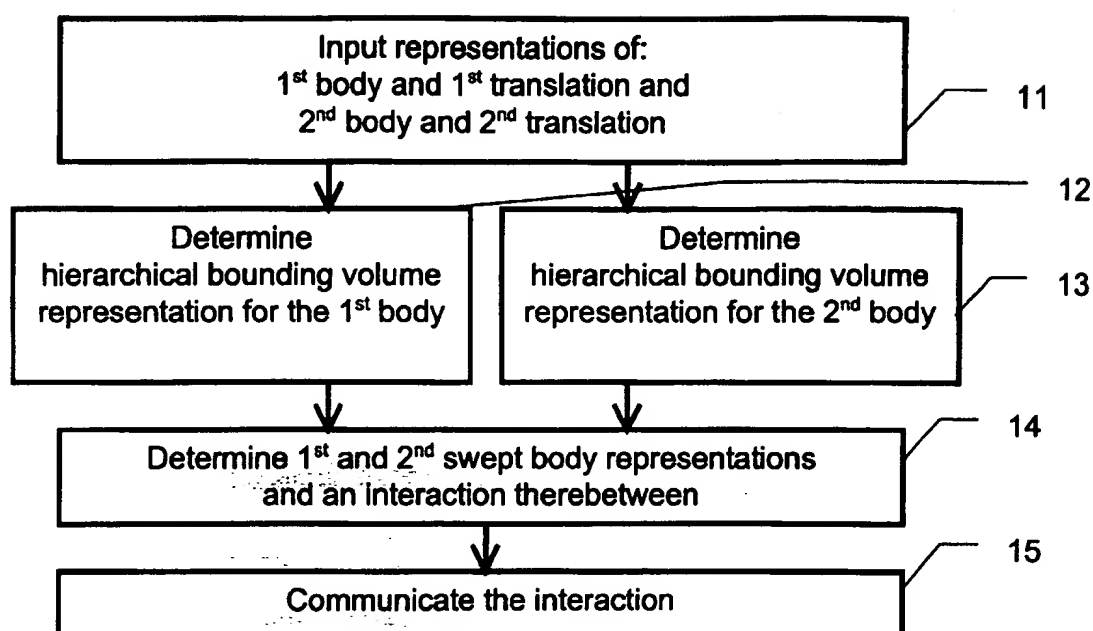


Figure 1

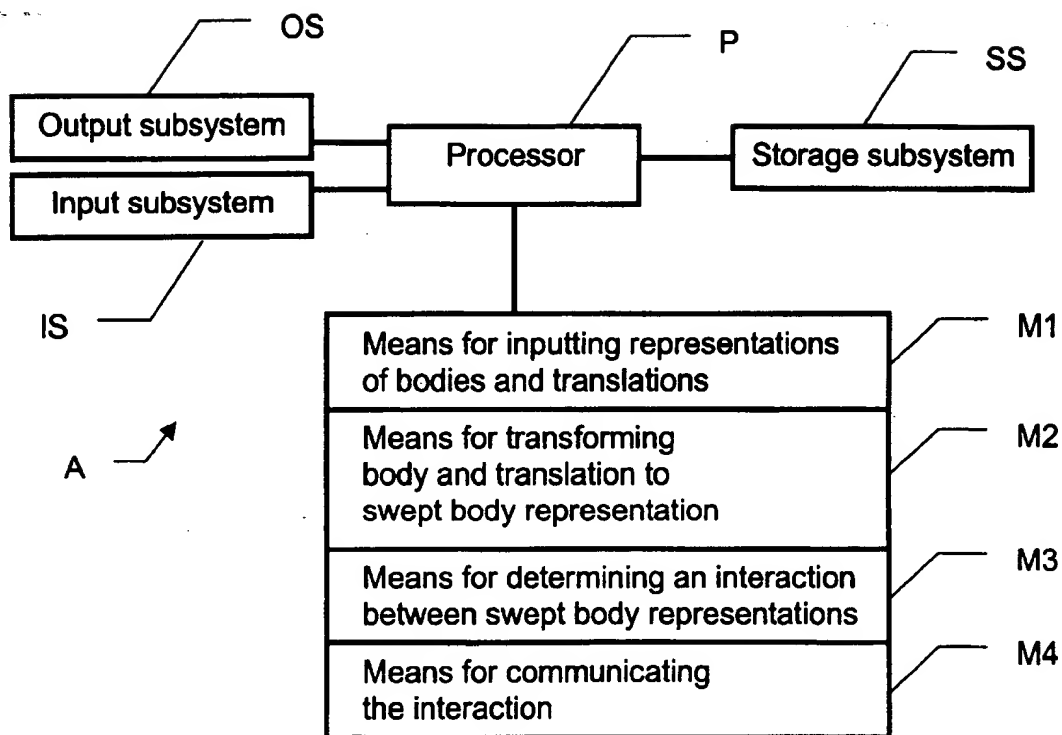


Figure 2

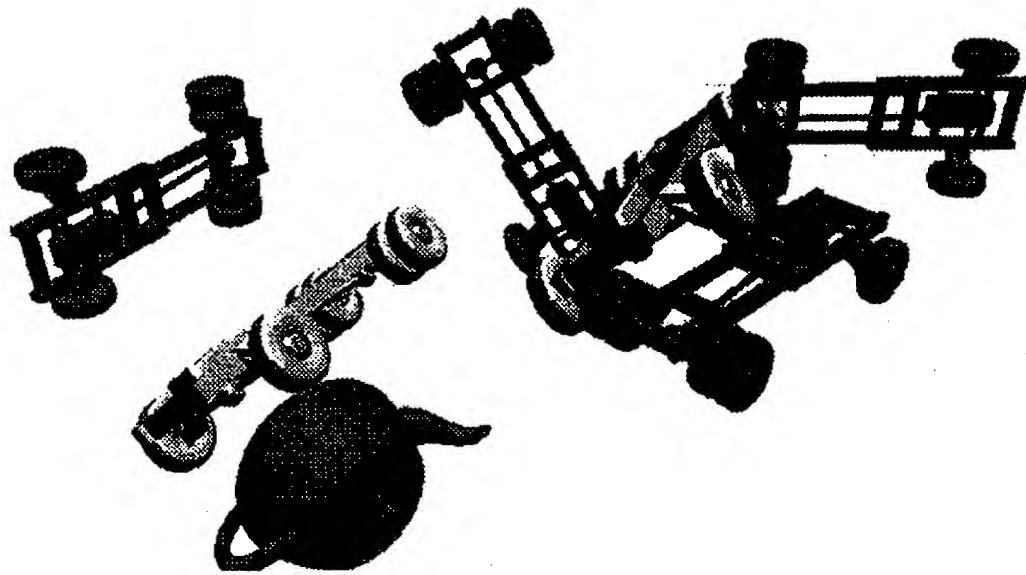


FIGURE 3

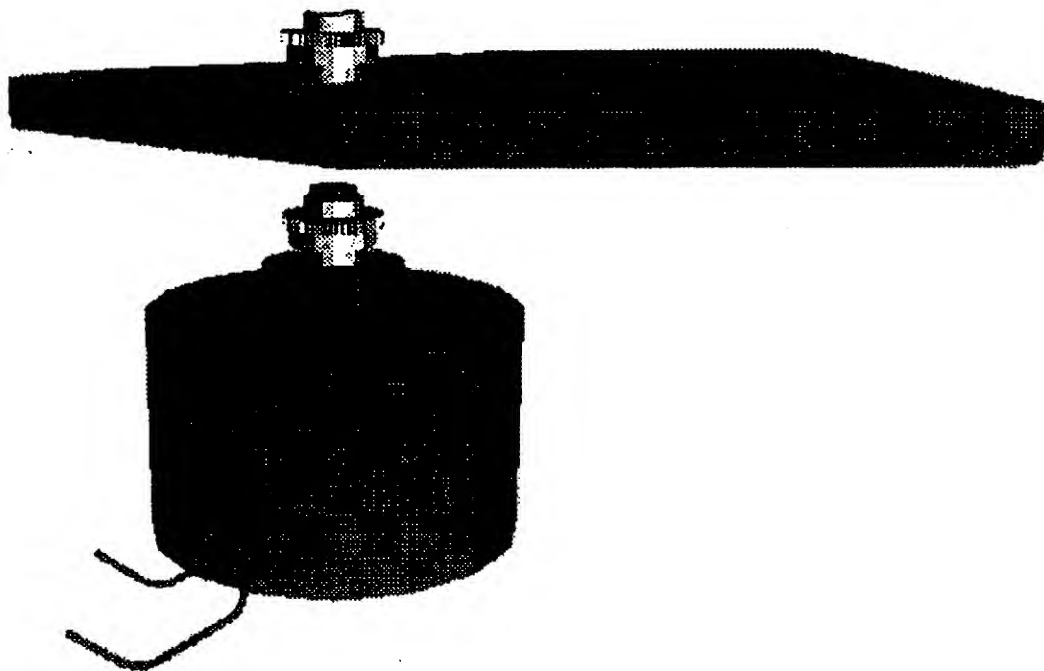


FIGURE 4

METHOD AND APPARATUS FOR MODELING INTERACTIONS

This invention was made with Government support under Contract DE-AC04-94AL85000 awarded by the U.S. Department of Energy. The Government has certain rights in the invention.

BACKGROUND OF THE INVENTION

This invention relates to the field of modeling interactions, specifically modeling interactions among complex polyhedra in motion.

Many robotics applications require modeling of interactions. For example, collision detection is a basic problem in robotics and related areas, arising in motion planning, control, graphical programming, motion-preview, virtual reality, and dynamical simulation. The collision detection problem asks whether a rigid body moving along a given path intersects with any of a set of obstacles at any point on that path. In a fuller version of the problem, all contacts must also be determined. In both cases, accuracy is of extreme importance when the results of collision detection between modeled objects affect the behavior of physical robots or influence the outcomes of physical simulations, such as those used in process and product design and evaluation.

Most current methods for collision detection rely on interference detection or distance computation. A few recent algorithms are very fast, using queries at steps in the bodies' motion. However, the difficulty of determining collisions or distances between steps mandates the use of very small step sizes, consequently reducing efficiency.

Interference detection (or collision detection) for a rigid body R at a position and orientation x and obstacles O means to determine whether $R(x) \cap O$ is non-empty, with $R(x)$ denoting the image of R in the world. Distance computation means to determine the minimum distance between all points in $R(x)$ and those in O . Simple collision detection for a rigid body R moving over a path segment among obstacles O means to determine whether at any point along the path the body contacts or intersects any obstacles. In other words, if we let C denote the space of positions and orientations and let $p: [0,1] \rightarrow C$ denote the path segment, then collision detection asks whether there is any $t \in [0,1]$ such that $R(p(t)) \cap O$ is non-empty.

Exact or accurate collision detection is often avoided for the sake of speed. Usually, either interference detection or distance computation queries are applied at closely spaced points on the path. Simple use of interference detection can miss collisions. Growing the objects by a safety margin can prevent this, but at the risk of detecting false collisions. With distance computation, one can vary the step size to reduce the number of queries needed to obtain a given resolution, but resolution is limited by the minimum step size.

For a number of years, hierarchical geometric representations have been used to avoid "all pairs" comparisons in interference detection, distance computation, and collision detection. The binary-space partitioning tree (BSPtree) and variants have been successfully used in exact interference checking and contact determination, but they do not readily yield distance information. See, e.g., Fuchs et al., "On visible surface generation by a priori tree structures," Proc. of ACM SIGGRAPH, pages 124-133, 1980. The octree is another space-partitioning data-structure that has been used in interference detection, but it must be used with other geometric objects to obtain computations matching the accuracy of faceted boundary representations or BSP-trees.

See, e.g., Jackins et al., "Octrees and their use in representing three-dimensional objects," Computer Graphics and Image Processing, (14):249-270, 1980; Meagher, "Geometric modeling using octree encoding," Computer Graphics and Image Processing, (19): 129-147, 1982.

Other hierarchical structures use various primitives to bound an object or its surface at each level of the hierarchy, although sometimes holes are treated explicitly. See, e.g., Faverion, "Hierarchical object models for efficient anti-collision algorithms," Proc. IEEE ICRA 1989, pages 333-340, Scottsdale, Ariz. 1989. Successful implementations for exact interference detection and distance computation have been based on several geometric primitives. Convex-hull based implementations, sphere-based techniques, and bounding-prism methods are among those known to be fast. See, e.g., Cameron, "Dealing with geometric complexity in motion planning," IEEE ICRA 1996 Workshop on Practical Motion Planning in Robotics, Minneapolis, Minn. 1996; Quinlan, "Efficient distance computation between non-convex objects," Proc. 1994 IEEE Int'l Conf on Robotics and Automation, San Diego, Calif. 1994; Sato et al., "Efficient collision detection using fast distance calculation algorithms for convex and non-convex objects," Proc. 1995 IEEE Int'l Conf on Robotics and Automation, pages 772-778, Minneapolis, Minn. April 1996; Gottschalk et al., "OBB-tree: A hierarchical structure for rapid interference detection," Proc ACM SIGGRAPH'90, 1996. However, none of these results cover swept-body distance or swept-body interference detection.

Canny described a technique for exact collision detection for convex polyhedra moving along path segments linear in translation and the quaternion representation of SO(3). See Canny, "Collision detection for moving polyhedra," IEEE Trans. on Pattern Analysis and Machine Intelligence, 8(2):200-209, 1986. Finally, Cameron used space-time techniques and develops algorithms and hierarchical structures for exact collision detection for non-convex objects. See Cameron, "Collision detection by 4D intersection testing," Int'l Journal of Robotics Research, 6(3):291-302, June 1990. Both these results are general in allowing exact time intervals of contact and penetration to be computed directly. Canny, however, only computes interactions between convex objects. Cameron describes a theoretical method for objects modeled by Boolean constructive solid geometry (CSG) trees and an implemented method of objects modeled by Boolean CSG trees with linear half-spaces at their leaves. The implemented method of Cameron is complicated and might be computationally expensive for a large class of non-convex objects, particularly those whose CSG models include difference operator. Neither Canny nor Cameron provides computation of the minimal clearance distance between bodies that do not collide.

Current technology thus provides two options, even when the motions are linear translations: (i) distance computation or interference detection between moving objects at discrete points in time; and (ii) exact collision detection in space-time without clearance distances that has limitations with regard to non-convex objects. The first option often requires very small motion steps for robustness (e.g., to not miss collisions), which causes a performance penalty in simulations, motion planning, graphical programming, and motion inspection. It is not an option when the motion step-size cannot be arbitrarily controlled, which is the case with currently available virtual reality motion-capture systems. Because many objects in the world are non-convex, the second option is of limited use in the applications mentioned above. Furthermore, the lack of robust minimal

clearance distance computation during motion limits what is feasible in safety analyses.

There is therefore a need for a method and apparatus for modeling interactions between moving bodies that is faster and more robust than current approaches.

SUMMARY OF THE INVENTION

The present invention provides a method and apparatus for modeling interactions that overcomes drawbacks of the prior approaches. The method of the present invention comprises representing two bodies undergoing translations by two hierarchical swept volume representations. Interactions such as nearest approach and collision can be modeled based on the swept body representations. The present invention is computationally fast and robust, and relatively simple to implement. It can serve as a practical tool in motion planning, CAD systems, simulation systems, safety analysis, and applications that require modeling time-based interactions.

A body can be represented in the present invention by a union of convex polygons and convex polyhedra. The body undergoing translation can then be represented by a swept body representation, where the swept body representation comprises a hierarchical bounding volume representation whose leaves each contain a representation of the region swept by a section of the body during the translation, and where the union of the regions is a superset of the region swept by the surface of the body during translation. Interactions between two bodies thus represented can be modeled by modeling interactions between the convex hulls of the finite sets of discrete points in the swept body representations.

Advantages and novel features will become apparent to those skilled in the art upon examination of the following description or may be learned by practice of the invention. The objects and advantages of the invention may be realized and attained by means of the instrumentalities and combinations particularly pointed out in the appended claims.

DESCRIPTION OF THE FIGURES

The accompanying drawings, which are incorporated into and form part of the specification, illustrate embodiments of the invention and, together with the description, serve to explain the principles of the invention.

FIG. 1 is a flow diagram of a method according to the present invention.

FIG. 2 is a schematic representation of an apparatus according to the present invention.

FIG. 3 is an example of interference determination according to the present invention.

FIG. 4 is an example of interference determination according to the present invention.

DETAILED DESCRIPTION OF THE INVENTION

The present invention provides a method for computing the minimum distance between the regions swept by two moving bodies that each experience an independent linear translation. If the two swept volumes interfere (i.e., the minimum distance is zero), the invention provides a method for detecting which pairs of primitives used in modeling the bodies give rise to this interference.

FIG. 1 is a flow diagram of a method according to the present invention. The method begins with input of repre-

sentations of first and second bodies and corresponding translations 11. The representations can be from computer-aided design (CAD) information, measured parameters from physical devices or models, or computer-predicted data, for example. Either or both translations can be null, i.e., the method models interactions between two bodies with either, both, or neither undergoing non-null translation.

Next, the input representations are transformed into hierarchical bounding volume representations 12, 13. One method suitable for producing hierarchical bounding volume representations is described by Xavier in "A Generic Algorithm for Constructing Hierarchical Representations of Geometric Objects," 1996 IEEE International Conference of Robotics and Automation, incorporated herein by reference. The hierarchy produced by the method of Xavier is a binary tree whose nodes each contain a convex polygon or convex polyhedron. The subtree rooted at a node represents the union of the primitives at its leaves. Thus, each node of Xavier's hierarchical geometric representation contains a conservative approximation, or wrapper, of the object represented by its subtree. One simple variation is to limit the number of vertices on each polyhedron or polygon at interior nodes of the bounding volume hierarchies, for example by replacing polyhedra and polygons in violation of the policy with minimal bounding prisms.

Those skilled in the art will appreciate that the bounding volume hierarchy for a given body need not be recomputed every time the body is used as input for the method of the present invention. Data structures can allow each body to be mapped to a different position and location in space for its starting points, simply by applying the respective mapping to the vertices in the respective bounding volume hierarchy.

Each node of the swept volume bounding volume hierarchy need only be computed as it is needed. The hierarchical swept volume representation is partially implicit in that at each node an actual bounding volume is not computed, but rather a set of vertices whose convex hull is the bounding volume is computed. At each node of the bounding volume representation we apply the motion transformation to the vertices (of the polygon or polyhedron) at the node and add the resulting vertices to those at the node; the convex hull of the combined set of vertices is the region swept out by the node, but we never have to compute this convex hull.

Interactions between the two bodies undergoing translations can then be determined from the swept body representations 14. For example, collisions can be determined. As another example, the distances between the bodies during translation can be modeled. The steps involved in determining certain of these interactions are presented in more detail below.

The interaction can then be communicated 15. For example, the interaction can be communicated directly to a user. As another example, the interaction can be communicated to a software tool for display or design feedback. Those skilled in the art will appreciate other communication modes and functions.

Distance Determination

A recursive method as in Table 1 can perform basic distance computation between two modeled bodies (without sweeps). At each stage the method considers a pair of nodes, one from each tree, beginning at the roots, with the distance dist set at infinity. In the base case, the method simply determines the distance between two convex primitives, and sets dist to this distance if it is smaller than the current value. In the recursive case, if the distance between the hulls of the

5

current nodes is no greater than dist, then the method recurses using the current node from one tree paired with each of the children of the current node from the other; otherwise, it cuts the current branch of the recursion. The len (b1)<len (b2) test is a heuristic, comparing principal axis lengths to decide which one of b1 and b2 should be descended in the recursive steps.

TABLE 1

```

real cstkDist(body *R, body *S)
{
    real dist = ∞;
    pairStack stack;
    body *b1, *b2;
    stack.push(R,S);
    while (!stack.isEmpty()) {
        stack.pop(&b1, &b2);
        if (isLeaf(b1) & isLeaf(b2))
            dist ← min(dist, primDist(b1,b2));
        else if (hullDist(b1,b2)>dist)
            continue;
        else if (isLeaf(b1) ∨ (isLeaf(b2) ∨ len(b1)<len(b2))) {
            stack.push(b1,b2→child1);
            stack.push(b1,b2→child2);
        }
        else {
            stack.push(b1→child1,b2);
            stack.push(b1→child2,b2);
        }
    }
    return dist;
}

```

Both hullDist(..) and primDist(..) determine the distance between two convex polyhedra (including convex polygons). This can be efficiently done according to the method known as Gilbert's Algorithm, described by Gilbert et al. in "A fast procedure for computing the distance between complex objects in three-dimensional space," IEEE Journal of Robotics and Automation, 4(2), April 1988, incorporated herein by reference.

To extend the method to the case of a translationally-swept bodies, note that Gilbert's Algorithm, given two finite sets of points, determines a minimum distance vector between their convex hulls. Observe that the convex hull of the vertices of a polyhedron (polygon) and their images under a translation x is equal to the swept hull of that polyhedron (polygon) under translation x. Thus, the method of the present invention can use Gilbert's Algorithm to determine the distance between the volumes swept by two polygons or polyhedra under linear translations. It does this by adding the new positions of the vertices to the lists of input vertices. Finally, if A, B, and C are convex and $A \cup B \subset C$, then $\text{sweep}(A,x) \cup \text{sweep}(B,x) \subset \text{sweep}(C,x)$, so the conservative approximation nature of the hierarchies is preserved under translational sweeps.

Accordingly, the method of the present invention modifies the method of Table 1 by first adding vectors x and y, which gives the translations to be applied to R and S, to the input parameter list. It then replaces the function calls primDist (b1, b2) and hullDist (b1, b2) with the call transGilbert(b1,x, b2, y), which calls Gilbert's Algorithm on the vertices of b1 unioned with their images under translation x, and on the vertices of b2 unioned with their images under translation y. The resulting method determines the exact distance between the volumes swept by b1 and b2 under translations x and y respectively.

As a further description of distance detection according to the present invention, consider the pseudocode shown in Table 2. This pseudocode takes two sets of vertices as input and computes the distance between the convex hulls of two

6

sets of vertices, which is the main work of primDist(..), hullDist(..), and transGilbert(..).

TABLE 2

```

real gilDist(vertex_set verts_1, vertex_set verts_2)
{
    real dist = ∞;
    simplex md_simplex = initial_Mink_diff_simplex(verts1, verts2);
    vector3D<real> dist_vector;
    repeat {
        if (simplex_spatially_contains(md_simplex, Origin))
            return(0);
        dist_vector = simplex_distance_vector(md_simplex, Origin);
        dist = length(dist_vector);
        if (verify_minimal(dist, dist_vector, verts_1, verts_2))
            return(dist);
        update_Mink_diff_simplex(dist_vector, &md_simplex,
                                verts_1, verts_2);
    }
}

```

The Table 2 pseudocode function gilDist(..) makes use of a simplex (md_simplex) from the Minkowski difference of the input sets of vertices verts_1 and verts_2. Here, a simplex is the convex hull of a single vertex, two distinct vertices, three non-co-linear vertices, or four non-co-planar vertices. This description assumes that a simplex is represented by its vertices. The Minkowski difference of two sets of points V and W, denoted "MinkowskiDiff(V, W)" is the set of all points p such that there exists some point v in V and some point w in W such that $p=v-w$. The distance between two convex hulls is equal to the distance from their Minkowski difference to the origin, and similarly, a vector is the minimal distance vector between two convex hulls if and only if it is the minimal distance vector between their Minkowski difference and the origin.

Once initialized, the simplex md_simplex always consists of vertices from MinkowskiDiff(verts_1, verts2). The following steps are then repeated until an answer is returned. If the current md_simplex overlaps the origin as determined by simplex_spatially_contains(..), then a distance of zero is returned. Otherwise, the simplex_distance_vector(..) computes the minimal-distance vector from md_simplex to the origin, and a function verify_minimal(..) determines whether this vector is also a minimal distance vector between the convex hulls of the two input sets of vertices. If it is, then its length is returned as the distance between the convex hulls. Otherwise, md_simplex is updated to a new simplex, i.e., one that the function has not considered in the current call to it, whose vertices are in MinkowskiDiff(verts_1, verts2). Gilbert gives one way of implementing simplex_distance_vector(..) as the "distance sub-algorithm". Those skilled in the art will appreciate other algebraically equivalent embodiments of the present invention.

If the step of modifying the simplex md_simplex in the method of Table 3 results in a simplex that is closer to the origin, then the method of Table 2 becomes a hillclimbing method, which is more efficient than exhaustive search. A step for modifying md_simplex that follows the method of Gilbert and meets this condition can be implemented as shown in the pseudocode of Table 3. This step is particularly effective at causing fast convergence and is recommended for use in the present invention; this is why we call the pseudocode function in Table 3 distGil(..).

TABLE 3

```

update_Mink_diff_simplex(vector3D<real> dist_vector,
                          simplex *md_simplex,
                          vertex_set verts_1, vertex_set vertex_2);

```

TABLE 3-continued

```

{ vertex vtx1, vtx2, splx_vrtx;
  vtx1 = max_support(dist_vector, verts_1);
  vtx2 = min_support(dist_vector, verts_2);
  splx_vrtx = vtx2 - vtx1;
  remove_extraneous(dist_vector, md_simplex);
  add_vertex(md_simplex, splx_vrtx);
}

```

The pseudocode in Table 3 assumes as inputs a vector `dist_vector`, a simplex `md_simplex`, and two sets of vertices `verts_1` and `verts_2`. `max_support(.)` takes a vector and a set of vertices as input and returns a vertex from this set furthest in the direction of the vector. `min_support(.)` takes a vector and a set of vertices as input and returns a vertex from this set furthest in the direction opposite the vector. `remove_extraneous(.)` takes a vector and a simplex and removes from the simplex those vertices that are not (tied for) furthest in the direction of the vector; if `update_Mink_diff_simplex(.)` is "called" as shown in the pseudocode for `gilDist(.)`, then the only vertices remaining lie in the plane that contains the point on the simplex that is closest to the origin and that is perpendicular to the vector from this point to the origin. `add_vertex(.)` simply adds a vertex to a simplex. Those skilled in the art will appreciate various methods for computing `min_support(.)` and `max_support(.)` that exploit different data structures for the input vertex set. Going through the input vertex set serially is sufficient, but performance can be improved when all the vertices in the input vertex set lie on the convex hull of the set, by storing edge-adjacency (on the convex hull) information applying and hillclimbing instead of a linear search. This latter method is asymptotically faster, taking roughly $O(\sqrt{n})$ time instead of $O(n)$ time, but has a larger asymptotic constant and requires a larger data structure. See e.g., Cameron, "Comparison of Two Fast Algorithms for Computing the Distance between Convex Polyhedra", published on the World Wide Web, September 1996. Because the planes passing through the origin and perpendicular to the edges of a vertex on a convex polyhedron determine the pre-image polytope of that vertex under `max_support(.)` or `min_support(.)`, it is possible to use well-known computational geometric methods to yield $O(\lg(n))$ implementations. This method, too has a larger asymptotic constant than the simple method and requires a more complex data structure. These methods can be extended to include the case in which the set of input vertices is the union of vertices of a number of known convex polyhedra (including polygons), most simply by computing the function for each of the convex polyhedra and then applying the serial version of the function to the resulting set of outputs.

The test performed by `verify_minimal(.)` merits additional description. There are two basic tests using an error bound `eps` based on the minimum accuracy in the machine representation of the vertices in the two bodies and applying standard numerical analysis techniques. First, if `dist` would be smaller than this bound `eps`, then `verify_minimal(.)` rounds `dist` to 0 and returns true. Second, `vrtx1` and `vrtx2` are computed as in Table 3, and from these two vertices and `dist_vector` we compute an error measure: `<dist_vector, dist13 vector>+<dist_vector, vrtx1>-<dist_vector, vrtx2>`. If this error measure is sufficiently small, then `verify_minimal(.)` returns true. Different assumptions yield different smallness criteria. In one simple criterion, this error measure is compared to the product of `eps` and `dist`. If the error measure is smaller, then `verify_minimal(.)` returns true. Finally, if the value of `dist` is not smaller than its

previous value, `verify_minimal(.)` returns true, and `dist` and `dist_vector` can revert to their previous values.

As a further description of distance determination according to the present invention, consider the pseudocode shown in Table 4. Here, we generalize the pseudocode from Table 1 to cover the case when bodies `R` and `S` are placed in configurations `conf_R` and `conf_S`, respectively, and then subjected to translations `x` and `y`, respectively, where a configuration for a rigid body is a particular position and orientation. We also show what should be done so that `distGil(.)` can be called directly, instead of using functions `primDist(.)`, `hullDist(.)`, and `transGilbert(.)` as described above.

TABLE 4

```

distance(body *R, configuration conf_R, vector x,
  body *S, configuration conf_S, vector y)
{
  real dist ← ∞;
  pairStack stack;
  body *b1, *b2;
  stack.push(R,S);
  while (!stack.isEmpty()) {
    stack.pop(&b1, &b2);
    verts1 = combine(b1->bounding_vertices(conf_R)),
      transMap(b1->bounding_vertices(conf_R), x);
    verts2 = combine(b2->bounding_vertices(conf_S)),
      transMap(b2->bounding_vertices(conf_S), y);
    if (isLeaf(b1) ^ isLeaf(b2))
      dist ← min(dist, gilDist(verts1, verts2));
    else if (gilDist(verts1, verts2) > dist)
      continue;
    else if (isLeaf(b1) v (isLeaf(b2) v len(b1) < len(b2)))
      foreach child in b2->children
        stack.push(b1, child);
    else
      foreach child in b1->children
        stack.push(child, b2);
  }
  return dist;
}

```

This pseudocode performs a tree traversal similar to that in Table 1. However, it assumes that the input bodies `R` and `S` are already bounding volume hierarchies, and that at each node of the hierarchy a set of bounding vertices whose convex hull bounds the union of the leaves of that node's subtree is available. This assumption is made because if the method will be used on a body more than once, with that body possibly at different configurations, it is disadvantageous to repeat construction of its bounding volume hierarchy over and over. This pseudocode does not assume that the hierarchy trees are binary, so the recursion must cover all the children of either the current `b1` or the current `b2`.

If `Bp` is a pointer to body `B` (which could be a node in a greater body), and `conf_B` is a configuration, then `Bp->bounding_vertices(conf_B)` returns a set of vertices whose convex hull bounds `B`. This can be implemented by simply mapping `B`'s bounding vertices to where they would be when `B` is in configuration `conf_B`. The particular set may have special properties that `max_support(.)` and `min_support(.)` can take advantage of, as described above. `transMap(.)` takes a vertex set and a translation and returns a copy of the set that has been moved by the translation. `combine(.)` takes two vertex sets as arguments and combines them into a set that is the union of the two sets and whose particular data structure preserves whatever properties that `min_support(.)` and `max_support(.)` can take advantage of. For example, if the support function can be computed in sublinear time individually on the two sets because of their data structure, then the data structure output by `combine` should allow the support function to be executed

individually on the two sets; the support function can then be run on the two outputs.

If x is a translation, then $\text{combine}(\text{Bp} \rightarrow \text{bounding_vertices}(\text{conf_B}))$, $\text{transMap}(\text{Bp} \rightarrow \text{bounding_vertices}(\text{conf_B}), x)$ is a set of bounding vertices for the region swept by B at starting at configuration conf_B and subjected to translation x . If B is a convex polygon or a convex polyhedron, then its vertices are its bounding vertices and the result of this combine is a set of vertices whose convex hull is exactly the swept region. Thus, since the $\text{distGil}(\cdot)$ function computes the distance between the convex hulls of its input vertex sets, the method of Table 4 implicitly transforms the input bodies into hierarchical bounding representations of their swept regions according to the input configurations and input translations.

Several optimizations can improve the performance of an implementation of distance (\cdot). One optimization is to, for each pair of bodies used as arguments to distance (\cdot), cache the pair of leaf nodes that yields the minimal distance; on the next call to the function for this pair, with the bodies possibly at different configuration, compute the distance between these two nodes in their new configurations, and then use this distance instead of infinity as the initial value of dist . A second optimization is to keep a cut-table of pairs of interior nodes for which it is believed that a check against the current value of dist could cause a cut in the recursion, and to do the pre-recursion checks ($\text{gilDist}(\text{verts1}, \text{verts2}) > \text{dist}$) only on the nodes in this table. Two additional optimizations apply to this implementation of $\text{distGil}(\cdot)$. First, add an optional max_dist argument, and cut short $\text{distGil}(\cdot)$'s computation if the actual distance exceeds this distance. When, distance (\cdot) calls $\text{distGil}(\cdot)$, this the current value of dist (in distance (\cdot)) is used for this argument. Second, distance (\cdot), $\text{distGil}(\cdot)$, and $\text{initial_Mink_diff_simplex}(\cdot)$ are modified so that the bounding vertices that give rise to the minimum-distance simplex found by $\text{distGil}(\cdot)$ are cached by distance (\cdot) for each pair of bodies ($b1, b2$) popped off the recursion stack stack , implementing a suggestion in Gilbert. As a matter of practicality, the cache is limited in size so that pairs are bumped from it when it gets full.

Collision Determination

The present invention accomplishes collision detection through a modification to the method of distance determination. A straightforward way to accomplish collision detection between two swept bodies is to compute the distance between them. If the resulting distance is smaller than a user-specified non-negative collision distance ("safety margin"), then the two swept bodies are determined to be in collision. As an optimization, distance (\cdot) can be modified to take such a safety margin as an additional argument; it initializes dist to this value instead of infinity. Finally, if the two bodies are moving translationally at constant velocities, swept-body collision detection can be used to determine whether they collide via transformation into a frame in which one of the bodies is stationary.

As a further example, consider the pseudocode in Table 5. The function $\text{collision}(\cdot)$ is similar to distance (\cdot). Instead of returning a distance, it returns a list of leaf node pairs whose swept regions are in collision, i.e., a witness list. The function takes two additional arguments: a collision distance (collision_dist) and the maximum length of witness list to return. In the base case, if $b1$ and $b2$ are primitives (leaf nodes), then we check whether the corresponding swept regions are at most collision_dist apart; if they are, we add them as a pair to the witness list. Also, collision_dist is used in the conditional instead of dist , as in the test for cutting

recursion. Obviously, actually accumulating a list would be optional if we only wanted a count of how many witnesses there are. Those knowledgeable in the art will appreciate that applicable optimizations described for distance (\cdot) and its calling subtree will not affect the correctness of collision (\cdot)

TABLE 5

```

list <prim_pair>
collisions(body *R, configuration conf_R, vector x,
           body *S, configuration conf_S, vector y,
           real collision_dist, int max_witnesses)
{
    list <prim_pair> collision_list(Empty);
    pairStack stack;
    body *b1, *b2;
    stack.push(R,S);
    while (!stack.isEmpty()) {
        stack.pop (&b1,&b2);
        verts1 = combine (b1->bounding_vertices(conf_R)),
            transMap(b1->bounding_vertices(conf_R), x);
        verts2 = combine(b2->bounding_vertices(conf_S)),
            transMap(b2->bounding_vertices(conf_S), y);
        if (isLeaf(b1) & isLeaf(b2)) {
            if (gilDist(verts1, verts2) <= collision_dist) {
                collision_list.enqueue(b1, b2);
                if (collision_list.size() >= max_witnesses)
                    return collision_list;
            }
        }
        else if (gilDist(verts1,verts2) > collision_dist)
            continue;
        else if (isLeaf(b1) & (!isLeaf(b2) & len(b1) < len(b2)))
            foreach child in b2->children
                stack.push(b1,child);
        else
            foreach child in b1->children
                stack.push(child,b2);
    }
    return collision_list;
}

```

Implementation

FIG. 2 shows an apparatus A according to the present invention. Processor P connects with input subsystem IS, output subsystem OS, and storage subsystem SS. Processor P can be, for example, a workstation such as those currently commercially available. Input subsystem IS can include, for example, a keyboard, a mouse, and a network connection. Output subsystem OS can include, for example, a video display, an audio speaker, a printer, and a network connection. Storage subsystem SS can include, for example, semiconductor memory, disk drives, tape storage, and network-based storage.

Apparatus A also comprises means for performing certain operations. These means can be programmed in software by techniques known to those skilled in the art. They can also be implemented in hardware, or in combination of hardware and software. Apparatus A includes means for using input subsystem IS to accept input representations of two bodies and corresponding translations M1. Apparatus A also includes means for transforming the input representations into swept body representations M2. Apparatus A also includes means for determining an interaction between the swept body representations M3. Apparatus A also includes means for using output subsystem OS for communicating the interaction M4.

Means for determining an interaction M3 can comprise means for determining closest approach or collision from volumes swept by nodes in bounding-volume hierarchical representations of two bodies. The means for determining an interaction can accordingly comprise means for determining distance between the convex hulls of two sets of vertices. The means for determining an interaction can accordingly comprise means for determining distance between the origin

and from the Minkowski difference of two sets of vertices. It can also comprise means for initializing the (vertices of) a simplex of the Minkowski difference of two sets of vertices. It can also comprise means for determining whether the simplex overlaps the origin. It can also comprise means for determining whether a line segment between a simplex and a point is a minimum-length line segment. It could also comprise means for modifying the simplex.

Example Result and Performance

Example of interference determination, implemented as distance determination subject to maximum and minimum thresholds, covered two model scenarios. In the first, shown in FIG. 3, coded "FT", a truck chassis (10K polygons) moves through an environment containing other truck chassis and a teapot (42K polygons total). The objects are up to 10 units in scale, and the translational path is 20 units long. Zero-distance occurs near both ends of the path, and the interference and distance queries are typical of gross-motion problems or free-flight.

In the second scenario, shown in FIG. 4, coded "GPM", a gear (3920 polygons) drops through a hole in a flat plate (4054 polygons) and onto the spindle of a motor (2252 polygons). The motor is roughly unit scale, and the translational path is one unit long. The clearance between the gear and the motor spindle is 0.00125 units. Except where noted, an initial call is always made to the distance function to initialize the caches, so that the runs reflect steady-state behavior.

The first series of experiments compared the costs of four types of queries: basic distance (D), interference detection (I), swept-body distance (DS), and swept-body interference detection (IS). For each model scenario, the motion was broken into 25-400 steps. The results are shown in Table 5. The table shows swept-body interference detection and distance computation costing up to 30-50 percent more time than their basic counterparts. Swept-body interference detection was typically half as expensive as basic distance computation.

An interesting observation is that, despite the much smaller polygon count, the GPM examples were computationally more intensive than the FT examples. Not only were clearances small in the hole and on the spindle, but in both cases, the geometric symmetry meant that there were many pairs of polygons approximately the minimal distance (0.00125) apart. This was a semi-pathological but realistic occurrence. While the gear moves onto the spindle in the last 20-25 percent of the motion, there are typically 500 pairs of polygons that are about the same, minimal distance apart, and distance calculation slowed to 10-15 Hz. For comparison, setting the interference threshold to 0.00128, the implementation exploits cached information to detect interference at over 2 kHz in that segment.

TABLE 5

scenario	num steps expt	25	50	100	200	400
time (seconds)						
FT	D	1.60	1.90	2.20	2.90	5.30
FT	I	0.32	0.49	0.83	1.50	2.70
FT	DS	1.50	1.90	2.50	3.80	6.40
FT	IS	0.44	0.64	1.10	2.00	3.50
GPM	D	2.00	3.30	5.90	10.70	20.90
GPM	I	0.59	1.10	1.70	3.10	6.00
GPM	DS	3.10	5.10	9.30	17.30	31.90
GPM	IS	1.00	1.60	2.60	4.60	8.60

Applications

Two important problems that arise in assembly planning are to determine: (i) whether a linear translation causes

interference, and (ii) which pairs of polygons interfere during the motion. Basic interference detection can only answer (i) and (ii) to a resolution equal to the step size. Accuracy is attainable if both the step size and the interference threshold (safety margin) can be precisely manipulated. Basic distance algorithms have the advantage of efficient step size estimates. Finally, both (i) and (ii) can also be reduced to polygon-polygon collision detection problems by stepping through the motion and collecting all polygon pairs less than a step size apart.

In contrast, a single swept-body interference query is sufficient to answer (i) exactly. On an implementation that can automatically collect all pairs of interfering polygons (i.e., witness pairs), a single query is also sufficient for (ii). Because the translational swept-body distance method is exact, it is not necessary to step through the motion. In the GPM scenario, the implementation of the present invention required 0.45 seconds each for (i) and (ii) when there was no contact ($\epsilon=0.00124$), and 0.45 seconds for (i) and 0.89 seconds (1335 witness pairs) for (ii) when there was contact ($\epsilon=0.00126$). For this test, no cache initialization queries were done, since it is likely that this sort of query would be done "cold" in practice.

Sequential-motion collision detection queries occur in dynamical simulation and other applications. An additional problem that arises is (iii) to determine contact initiation and termination points during a timestep. Computing a conservative approximation of (ii)—finding a guaranteed superset of polygon pairs that interfere sometime during a step—reduces this problem to a collection of polygon-polygon problems.

For an example of this sub-task, two cases of the gear-plate-motor example were considered, one in which the 0.00125 unit clearance is considered interference/collision ($\epsilon=0.00128$), and one in which it is not. Five experiments were conducted. Using basic thresholded interference detection with witness collection, we set the interference threshold to half the step size (TW) and to the same value plus ϵ (TW ϵ). Swept-body interference detection with witness collection was tested both with the ϵ (ISW ϵ) and without (ISW). The swept-body tests used a hybrid implementation—the swept-body code was triggered when basic code would collect witnesses because of the interference threshold. This accounts for GPM/ISW being faster than GPM/IS. The swept-body interference detection with the E but without witness collection (IS ϵ) was also performed.

TABLE 6

scenario	num steps expt	25	50	100	200	400
time (seconds)						
GPM	TW	3.70	3.60	5.70	9.80	7.80
GPM	TW ϵ	3.70	3.70	5.80	9.40	17.80
GPM	ISW	0.83	1.20	2.10	3.50	7.80
GPM	IS ϵ	0.44	0.60	0.84	1.49	2.85
GPM	ISW ϵ	1.90	3.00	5.10	8.70	16.80

Two important comparisons are evident. First, compare GPM/ISW and GPM/IS ϵ against GPM/TW and GPM/TW ϵ to consider how useful swept-body interference detection might be in answering problem (i)—determining whether a collision takes place during a motion step. The data shows the swept-body method to be much faster except when there is no interference and the step size is smaller than the clearance. To consider efficiency in collecting witnesses for accurately answering problems (ii) and (iii), compare GPM/ISW and GPM/ISW ϵ to GPM/TW and GPM/TW ϵ . The basic code is several times as costly as the swept-body code

13

when the former must collect many polygon pairs that the latter safely rules out, and that the two methods cost about the same when they find similarly high numbers of collision candidates.

The particular sizes and equipment discussed above are cited merely to illustrate particular embodiments of the invention. It is contemplated that the use of the invention may involve components having different sizes and characteristics. It is intended that the scope of the invention be defined by the claims appended hereto.

I claim:

1. A method for using a computer to model interactions between a first body experiencing a first translation and a second body experiencing a second translation, comprising:

- a) receiving into the computer representations of said first body, said first translation, said second body, and said second translation;
- b) using the computer to transform the representation of said first body and the representation of said first translation into a first swept representation comprising a hierarchical bounding volume representation whose leaves each contain a representation of the region swept by a section of said first body during said first translation, wherein the union of said volumes is a superset of the region swept by the surface of said first body during said first translation;
- c) using the computer to transform the representation of said second body and the representation of said second translation into a second swept representation comprising a hierarchical bounding volume representation whose leaves each contain a representation of the region swept by a section of said second body during said second translation, wherein the union of said volumes is a superset of the region swept by the surface of said second body during said second translation;
- d) using the computer to determine an interaction between said first body and said second body; and
- e) communicating the interaction.

2. A method according to claim 1, wherein the interaction determined is the minimum distance between said first body and said second body.

3. A method according to claim 2, wherein the representations of said first body and said second body comprise unions of convex polygons and convex polyhedra.

4. A method according to claim 3, wherein the step of using the computer to determine the minimum distance comprises using the computer to determine the minimum distance between the convex hull of a first finite set of discrete points in said first swept representation and the convex hull of a second finite set of discrete points in said second representation.

5. A method according to claim 4, wherein the step of using the computer to determine the minimum distance comprises using the computer according to Gilbert's Algorithm using a first set of points in said first swept representation and a second set of points in said second representation as inputs to Gilbert's Algorithm.

6. A method according to claim 4, wherein said discrete points are represented in a coordinate system having an origin, and wherein the step of using the computer to determine the minimum distance between the convex hull of a first finite set of discrete points and the convex hull of a second finite set of discrete points comprises:

- a) initializing the vertices of a simplex using points from the Minkowski difference set of said first set of points and said second set of points;

14

- b) determining whether said simplex overlaps said origin;
- c) determining whether a line-segment between said simplex and said origin has its non-origin vertex on the convex hull of said Minkowski difference set and is a minimum-length line-segment from said hull to the origin, and if so, determining the minimum length to be the length of said line segment;

- d) otherwise, modifying said simplex using vertices from said Minkowski difference set and repeating step (b) through (d).

7. A method according to claim 4, wherein said discrete points are represented in a coordinate system having an origin, and wherein the step of using the computer to determine the minimum distance between the convex hull of a first finite set of discrete points and the convex hull of a second finite set of discrete points comprises:

- a) initializing the vertices of a first simplex from the Minkowski difference set of said first set of points and said second set of points;
- b) determining a current distance vector as a minimum-length vector from said first simplex to said origin;
- c) if the length of said current distance vector is less than a minimum length, then determining said minimum distance to be zero;
- d) otherwise, determining an error measure from said current distance vector, said first set of points, and said second set of points, and if said error measure is sufficiently small then determining said minimum distance to be the length of said current distance vector;
- e) otherwise, if possible, forming a modified simplex from said first simplex by removing all vertices from said first simplex that do not lie on the plane perpendicular to the current distance vector and passing through a point on said first simplex closest to said origin, and by adding a closer vertex from the Minkowski difference set, wherein a closer vertex is a vertex further in the direction of said current distance vector, and repeating steps (b) through (e) using said modified simplex in place of said first simplex;
- f) otherwise, determining said minimum distance to be the length of the current distance vector.

8. A method according to claim 1 wherein the interaction determined is collision between said first body and said second body.

9. A method according to claim 8 wherein the first and second bodies are represented by unions of convex polygons and convex polyhedra.

10. A method according to claim 9 wherein the step of using the computer to determine collision comprises using the computer to determine collision between the convex hull of a first finite set of discrete points in said first swept representation and the convex hull of a second finite set of discrete points in said second representation.

11. A method according to claim 10 wherein the step of using the computer to determine collision comprises using the computer according to Gilbert's Algorithm using a first set of points in said first swept representation and a second set of points in said second representation as inputs to Gilbert's Algorithm.

12. A method according to claim 10, wherein said discrete points are represented in a coordinate system having an origin, and wherein the step of using the computer to determine collision between the convex hull of a first finite set of discrete points and the convex hull of a second finite set of discrete points comprises:

- a) initializing the vertices of a simplex using points from the Minkowski difference set of said first set of points and said second set of points;

15

- b) determining whether said simplex overlaps said origin, and if so, determining that a collision occurs;
- c) determining whether a line-segment between said simplex and said origin has its non-origin vertex on the convex hull of said Minkowski difference set and is a minimum-length line-segment from said convex hull to said origin, and if so, determining whether the length of said line segment is less than a minimum length, and if so, determining that a collision occurs;
- d) otherwise, modifying said simplex using vertices from said Minkowski difference set and repeating step (b) through (d);
- e) otherwise, determining that no collision occurs.

13. A method according to claim 10, wherein said discrete points are represented in a coordinate system having an origin, and wherein the step of using the computer to determine collision between the convex hull of a first finite set of discrete points and the convex hull of a second finite set of discrete points comprises:

- a) initializing the vertices of a first simplex from the Minkowski difference set of said first set of points and said second set of points;
- b) determining a current distance vector as a minimum-length vector from said first simplex to said origin;
- c) if the length of said current distance vector is less than a minimum length, then determining that collision occurs;
- d) otherwise, determining an error measure from said current distance vector, said first set of points, and said second set of points, and if said error measure is sufficiently small then if the length of said current distance vector is less than a minimum length the determining that collision occurs, and if the length of said current distance vector is not less than a minimum length then determining that a collision does not occur;
- e) otherwise, if possible, forming a modified simplex from said first simplex by removing all vertices from said first simplex that do not lie on the plane perpendicular to the current distance vector and passing through a point on said first simplex closest to the origin, and by adding a closer vertex from the Minkowski difference set, wherein a closer vertex is a vertex further in the direction of said current distance vector, and repeating steps (b) through (d) using said modified simplex in place of said first simplex;
- f) otherwise, determining that no collision occurs.

14. An apparatus for modeling interactions between a first body experiencing a first translation and a second body experiencing a second translation, comprising:

- a) a processor;
- b) a storage subsystem;
- c) an input subsystem;
- d) an output subsystem;
- e) means for using said input subsystem to receive into said storage subsystem representations of said first body, said first translation, said second body, and said second translation;
- f) means for transforming said representation of said first body and said representation of said first translation into a first swept representation comprising a hierarchical bounding volume representation whose leaves each contain a representation of the region swept by a section of said first body during said first translation, wherein the union of said volumes is a superset of the

16

region swept by the surface of said first body during said first translation;

- g) means for transforming the representation of said second body and said representation of said second translation into a second swept representation comprising a hierarchical bounding volume representation whose leaves each contain a representation of the region swept by a section of said second body during said second translation, wherein the union of said volumes is a superset of the region swept by the surface of said second body during said second translation;
- h) means for determining an interaction between said first body and said second body; and
- i) means for using said output subsystem to communicate said interaction.

15. The apparatus of claim 14, wherein the means for determining an interaction comprise means for determining the minimum distance between said first swept representation and said second swept representation.

16. The apparatus of claim 15 wherein the means for determining the minimum distance comprises means for determining the minimum distance between the convex hull of a first finite set of discrete points in said first swept representation and the convex hull of a second finite set of discrete points in said second representation.

17. The apparatus of claim 16, wherein said discrete points are represented in a coordinate system having an origin, and wherein the means for determining the minimum distance between the convex hull of a first finite set of discrete points and the convex hull of a second finite set of discrete points comprises:

- a) means for initializing the vertices of a simplex using points from the Minkowski difference set of said first set of points and said second set of points;
- b) means for determining whether said simplex overlaps said origin;
- c) means for determining whether a line-segment between said simplex and said origin has its non-origin vertex on the convex hull of said Minkowski difference set and is a minimum-length line-segment from said convex hull to said origin, and if so, determining the minimum length to be the length of said line segment;
- d) means for modifying said simplex using vertices from said Minkowski difference set and repeating step (b) through (d).

18. The apparatus of claim 16, wherein said discrete points are represented in a coordinate system having an origin, and wherein the means for determining the minimum distance between the convex hull of a first finite set of discrete points and the convex hull of a second finite set of discrete points comprises:

- a) means for initializing the vertices of a first simplex from the Minkowski difference set of said first set of points and said second set of points;
- b) means for determining a current distance vector as a minimum-length vector from said first simplex to said origin;
- c) means for determining whether the length of said current distance vector is less than a minimum length, and if so, then determining said minimum distance to be zero;
- d) means for determining an error measure from said current distance vector, said first set of points, and said second set of points, and if said error measure is sufficiently small then determining said minimum distance to be the length of said current distance vector;

17

e) means for, if possible, forming a modified simplex from said first simplex by removing all vertices from said first simplex that do not lie on the plane perpendicular to the current distance vector and passing through a point on said first simplex closest to said origin, and by adding a closer vertex from the Minkowski difference set, wherein a closer vertex is a vertex further in the direction of said current distance vector, and repeating steps (b) through (e) using said modified simplex in place of said first simplex;

f) means for determining said minimum distance to be the length of the current distance vector if it is not possible to form a modified simplex according to (e).

19. The apparatus of claim 14, wherein the means for determining an interaction comprise means for determining collision between said first body and said second body.

20. The apparatus of claim 19 wherein the means for determining collision comprises means for determining the collision between the convex hull of a first finite set of discrete points in said first swept representation and the convex hull of a second finite set of discrete points in said second representation.

21. The apparatus of claim 20, wherein said discrete points are represented in a coordinate system having an origin, and wherein the means for determining collision between the convex hull of a first finite set of discrete points and the convex hull of a second finite set of discrete points comprises:

- a) means for initializing the vertices of a simplex using points from the Minkowski difference set of said first set of points and said second set of points;
- b) means for determining whether said simplex overlaps said origin, and if so, determining that a collision occurs;
- c) means for determining whether a line-segment between said simplex and said origin has its non-origin vertex on the convex hull of said Minkowski difference set and is a minimum-length line-segment from said convex hull to said origin, and if so, determining whether the length of said line segment is less than a minimum length, and if so, determining that a collision occurs;

18

d) means for, if possible, modifying said simplex using vertices from said Minkowski difference set and repeating step (b) through (d);

e) means for determining that no collision occurs if it is not possible to modify said simplex according to (d).

22. The apparatus of claim 20, wherein said discrete points are represented in a coordinate system having an origin, and wherein the means for determining collision between the convex hull of a first finite set of discrete points and the convex hull of a second finite set of discrete points comprises:

- a) means for initializing the vertices of a first simplex from the Minkowski difference set of said first set of points and said second set of points;
- b) means for determining a current distance vector as a minimum-length vector from said simplex to said origin;
- c) means for determining that collision occurs if the length of said current distance vector is less than a minimum length;
- d) means for determining an error measure from said current distance vector, said first set of points, and said second set of points, and if said error measure is sufficiently small then if the length of said current distance vector is less than a minimum length the determining that collision occurs, and if the length of said current distance vector is not less than a minimum length then determining that a collision does not occur;
- e) means for if possible, forming a modified simplex from said first simplex by removing all vertices from said first simplex that do not lie on the plane perpendicular to the current distance vector and passing through a point on said first simplex closest to said origin, and by adding a closer vertex from the Minkowski difference set, wherein a closer vertex is a vertex further in the direction of said current distance vector, and repeating steps (b) through (d) using said modified simplex in place of said first simplex;
- f) means for determining said minimum distance to be the length of the current distance vector if it is not possible to form a modified simplex according to (e).

* * * * *